NASA Procedural Requirements

NPR 7150.2
Effective Date: September 27, 2004
Expiration Date: September 27, 2009

COMPLIANCE IS MANDATORY

Printable Format (PDF)

Subject: NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

CHAPTER 3: Software Engineering (Life Cycle) Requirements

This NPR makes no recommendation for a specific life cycle model. Each has its strengths and weaknesses, and no one model is best for all situations. Whether using the spiral model, the iterative model, waterfall, or any other development life cycle model, each has steps of requirements, design, implementation, testing, release to operations, maintenance, and retirement. Without recommending a life cycle, the requirements for each of these steps are provided below.

3.1 Software Requirements

The requirements phase is one of the most important phases of software engineering. Studies show that the top problems in the software industry are due to poor requirements elicitation, inadequate requirements specification, and inadequate management of changes to requirements. Requirements provide the foundation for the entire life cycle as well as for the software product. Requirements also provide a basis for planning and estimating. Requirements are based on customer, user, and other stakeholder needs and design and development constraints. The development of requirements includes elicitation, analysis, documentation, verification, and validation. It is important that there is ongoing customer validation of the requirements to ensure the end products meet the customer needs. This can be accomplished via rapid prototyping and customer involved reviews of iterative and final software requirements.

3.1.1 Requirements Development

3.1.1.1 The project shall document the software requirements. [SWE-049]

Note: The requirements for the content of each Software Requirement Specification and Data Dictionary are defined in Chapter 5.

3.1.1.2 The project shall identify, develop, document, approve, and maintain software requirements based on analysis of customer and other stakeholder requirements and the operational concepts. [SWE-050]

3.1.1.3 The project shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements and the hardware specifications and design. [SWE-051]

Note: This analysis is for safety criticality, correctness, consistency, clarity, completeness, traceability, feasibility, verifiability, and maintainability. This includes the allocation of functional and performance requirements to functions and subfunctions.

3.1.1.4 The project shall perform, document, and maintain bi- directional traceability between the software requirement and the higher level requirement. [SWE-052]

Note: The project should identify any orphaned or widowed requirements (no parent or no child) associated with reused software.

3.1.2 Requirements Management

3.1.2.1 The project shall collect and manage changes to the software requirements. [SWE-053]

Note: The project should analyze and document changes to requirements for cost, technical, and schedule impacts.

3.1.2.2 The project shall identify inconsistencies between requirements, project plans, and software products and initiate corrective actions. [SWE-054]

Note: A verification matrix supports the accomplishment of this requirement.

3.1.2.3 The project shall perform requirements validation to ensure that the software will perform as intended in the customer environment. [SWE-055]

Note: This should include confirmation that the requirements meet the needs and expectations of the customer.

3.2 Software Design

Architectural design is concerned with creating a strong overall structure for software entities that fulfill allocated system and software- level requirements. Typical views captured in an architectural design include the decomposition of the software subsystem into design entities, computer software configuration items (CSCI), definitions of external and internal interfaces, dependency relationships among entities and system resources, and finite state machines. Detailed design further refines the design into lower level entities that permit the implementation by coding in a programming language. Typical attributes that are documented for lower level entities include: identifier, type, purpose, function, constraints, subordinates, dependencies, interface, resources, processing, and data. Rigorous specification languages, graphical representations, and related tools have been developed to support the evaluation of critical properties at the design level. Projects are encouraged to take advantage of these improved design techniques to prevent and eliminate errors as early in the life cycle as possible.

3.2.1 The project shall document the software design. [SWE-056]

Note: The requirements for the content of the software design description and interface design description are defined in Chapter 5.

3.2.2 The project shall transform the allocated and derived requirements into a documented architectural design. [SWE-057]

3.2.3 The project shall develop and record a detailed design based on the architectural design that describes the lower level units so that they can be coded, compiled, and tested. [SWE-058]

3.2.4 The project shall perform and maintain bi-directional traceability between the software requirements and the software design. [SWE-059]

3.3 Software Implementation

Software implementation consists of implementing the requirements and design into code, data, and documentation. Software implementation also consists of following coding methods and standards. Unit testing is also usually a part of software implementation (unit testing can also be conducted during the testing phase).

3.3.1 The project shall implement the software design into software code. [SWE-060]

3.3.2 The project shall ensure that software coding methods, standards, and/or criteria are adhered to and verified. [SWE-061]

3.3.3 The project shall ensure that the software code is unit tested per the plans for software testing. [SWE-062]

3.3.4 The project shall provide a Software Version Description document for each software release. [SWE-063]

Note: The requirements for the content of the Software Version Description document are defined in Chapter 5.

3.3.5 The project shall provide and maintain traceability from software design to the software code. [SWE-064]

3.4 Software Testing

The purpose of testing is to verify the software and remove defects. Testing verifies the code against the requirements and the design to ensure that the requirements are implemented. Testing also identifies problems and defects that are corrected and tracked to closure before product delivery. Testing should also validate that the software operates appropriately in the intended environment.

3.4.1 The project shall provide: [SWE-065]
a. Software Test Plan(s).
b. Software Test Procedures.

c. Software Test Reports.

Note: The requirements for the content of the Software Test Plan, Software Test Procedures, and Software Test Reports are defined in Chapter 5.

3.4.2 The project shall perform software testing as defined in the Software Test Plan. [SWE-066]

Note: Testing could include software integration testing, systems integration testing, end-to-end testing, acceptance testing, white and black box testing, decision and path analysis, statistical testing, stress testing, performance testing, regression testing, qualification testing, simulation, and others. Automated testing tools should also be considered.

3.4.3 The project shall ensure that the implementation of each software requirement is verified to the requirement. [SWE-067]

3.4.4 The project shall evaluate test results and document the evaluation. [SWE-068]

3.4.5 The project shall document defects identified during testing and track to closure. [SWE-069]

3.4.6 The project shall test, validate, and certify software models, simulations, and analysis tools. [SWE-070]

3.4.7 The project shall update Software Test Plan(s) and Software Test Procedure(s) to be consistent with software requirements. [SWE-071]

3.4.8 The project shall provide and maintain traceability from the Software Test Procedures to the software requirements. [SWE-072]

3.4.9 The project shall ensure that the software system is validated on the targeted platform or high-fidelity simulation. [SWE-073]

3.5 Software Operations, Maintenance, and Retirement

Planning for operations, maintenance, and retirement begins early in the software life cycle. Operational concepts and scenarios are derived from customer requirements and validated in the operational or simulated environment. Software maintenance activities sustain the software product after it is delivered to the customer until retirement.

3.5.1 The project shall document the software maintenance plans in the Software Maintenance Plan document. [SWE-074]

Note: The requirements for the content of the Software Maintenance Plan are defined in Chapter 5.

3.5.2 The project shall plan software operations, maintenance, and retirement activities. [SWE-075]

3.5.3 The project shall implement software operations, maintenance, and retirement activities as defined in the respective plans. [SWE-076]

3.5.4 The project shall complete and deliver the software product to the customer with appropriate documentation to support the operations and maintenance phase of the software life cycle. [SWE-077]

Note: Delivery includes, as applicable, Software User's Manual (as defined in Chapter 5), source files, executable software, procedures for creating executable software, procedures for modifying the software, and a Software Version Description. Open source software licenses should be reviewed by the Center Chief of Patent/Intellectual Property Counsel before being accepted into software development projects. Other documentation that should be considered for delivery is:
a. Summary and status of all accepted Change Requests to the baselined Software Requirements Specifications.
b. Summary and status of all major software capability changes since baselining of the Software Design Documents.
c. Summary and status of all major software tests (including development, verification, and performance testing).
d. Summary and status of all Discrepancy Reports written against the software.
e. Summary and status of all software requirements deviations and waivers.
f. Summary and status of all software user notes.
g. Summary and status of all quality measures historically and for this software.
h. Definition of open work, if any.
i. Software configuration records defining the verified and validated software, including requirements verification data (e.g., requirements verification matrix).
j. Final version of the software documentation, including the final Software Version Description document(s).
k. Summary and status of any open software-related risks.

3.5.5 The project shall deliver to the customer the as-built documentation to support the operations and maintenance phase of the software life cycle. [SWE-078]

DISTRIBUTION:
NODIS